

Converting Alliance EDIF to Lattice EDIF

N.J.Gunton 3/00, 10/00, 10/02, 11/03

Grouping	Individual / Teams
Prerequisites	An understanding of VHDL, Linux & Alliance toolkit.
Courses	CRTS, CSE, DSE, EE, any interested party
Requirements	Linux system running Glibc2.1 & NT running Lattice synthesis software. Security keys, download cable & target board.
Summary	An introduction to the conversion of EDIF files
Objectives	To enable a student to tackle the conversion of vhdl files to Alliance EDIF and then to Lattice EDIF. To gain an understanding of the structure of EDIF files
Presuppositions	Assumes you are running bash or sh as your shell

Overview:

EDIF, or Electronic Design Interchange Format, is an industry standard for the exchange of data describing electronic circuits. (URL..<http://www.edif.org>) The reasoning behind the format is to simplify the transfer of circuit designs between different tools. Unfortunately it doesn't quite work as easily as one might like and some editing is almost always required.

"An ASIC designer spends an increasing amount of time forcing different tools to communicate ... The structure of EDIF is similar to Lisp or Postscript. This makes EDIF a very hard language to read and almost impossible to write by hand. EDIF is intended as an exchange format between tools, not as a design entry language. Since EDIF is so flexible each company reads and writes different 'flavours' of EDIF. Inevitably EDIF from one company does not quite work when we try and use it with tools from another company. ... We need to know just enough about EDIF to be able to fix these problems."

M.J.S. Smith in Application-Specific Integrated Circuits

This worksheet assumes that you are starting with a .vbe file. This can be a data-flow description that you have written yourself or one that has been generated for you by `syf`. It is assumed that you have tested the behaviour of this file using the simulator and are happy with the results. If you don't have an existing file then there is an example file `t_bird.fsm` in `/usr/local/alliance/examples`. for you to use. Note that you will have to convert it to .vbe and test it using the provided test patterns (also in the examples directory). There is also a hand-written .vbe file, `t_bird_hand.vbe`

Phase 1:

• NOTE:

At some point the Lattice libraries (`latlib`) and the tools `al2lat`, `edi2edf`, `lastpart`, and `sxconv` will be moved to `/usr/local/alliance/cells` and `/usr/local/alliance/bin` respectively. In the meantime use the paths referred to below. The man page for `sxconv` will also be placed in a sensible location.

- Check the value of the environment variables `MBK_TARGET_LIB` and `MBK_CATA_LIB`. For this phase of the operation `MBK_TARGET_LIB` should be set to `/$ALLIANCE_TOP/examples/Convert/latlib`. `MBK_CATA_LIB` should also be set to the same path. To check the environment variables use the command `printenv`. This will write all the currently set variables to screen. If there is no mention of any of the above variables then you will need to

set them to their default values first. This can be done on the command line as follows. Note that the leading 'dot' 'space' is significant.

```
. /usr/local/alliance/etc/alc_env.sh
```

It is probably worthwhile taking time out to edit your `.bash_profile` file and adding the above line to it, as well as the following resetting of `MBK_CATA_LIB`. The required variables will then be set at login. To set or alter the variables use the command `export`¹. For example to add `latlib` to `MBK_CATA_LIB` do (*but don't type the backslash*)

```
export MBK_CATA_LIB=/usr/local/alliance/examples/Convert/latlib:/usr/local/\
alliance/cells/sxlib
```

and to change the target library to `latlib`

```
export MBK_TARGET_LIB=/usr/local/alliance/examples/Convert/latlib
```

Now convert your `.vbe` file to a `.vst` file by the use of the tool `boog`. The syntax for the command is

```
boog inputfile
```

The file extensions are left off, as per the other Alliance tools, the output file name is optional, the default will be to use the same as the input filename, the extension will be `.vst`

If you get an error about "Vdd and Vss are needed in port of file ..." then you need to add the power inputs to your data-fw (`.vbe`) model or your state-machine model (`.fsm`) as appropriate.

Phase 2:

Having got your structural description, you now need to convert it into an EDIF file.

- First check that you have set `MBK_TARGET_LIB` to the lattice libraries `/usr/local/alliance/examples/Convert/latlib`.
- Run the command `/usr/local/alliance/examples/Convert/al2lat` to convert the cell names in your `.vst` file to the lattice cell names. You will need to use file redirection for this at present.

```
al2lat <inputfile.vst >outputfile.vst
```

Note that the output file MUST have a different name to the input file but the same extension. This will create a structural description using the Lattice library names. This now has to be converted into an EDIF description. The following command will convert your file from `vst` to `edi`. The format is `input-file-type output-file-type input-file output-file`.

You must set `MBK_CATA_LIB` to use `latlib` for this stage as well as the target library.

```
x2y vst edi in-file out-file
```

All being well you should have an EDIF description of your design.

Phase 3:

The EDIF file output by the Alliance toolkit needs to be edited before it can be accepted by the Lattice fitter. The editing is straightforward and can be done using `emacs`. Inspection of the EDIF file will show that it is structured similarly to a VHDL structural file. An extract is given below.

¹ If using the `bash` shell that is. For `csh` and similar do `set MBK_TARGET_LIB /usr/local/alliance/examples/Convert/latlib`. If in doubt ask a *nix guru :)

```
(cell and2 (cellType GENERIC)
(view schematic (viewType NETLIST)
(interface
(port z0 (direction OUTPUT))
(port a1 (direction INPUT))
(port a0 (direction INPUT)))
(contents
(net z0 (joined
(portref z0 )))
(net a1 (joined
(portref a1 )))
(net a0 (joined
(portref a0 ))))))
```

```
(cell and2 (cellType GENERIC)
(view symbol (viewType NETLIST)
(interface
(port z0 (direction OUTPUT))
(port a1 (direction INPUT))
(port a0 (direction INPUT))))
```

Inspection of this code will show that there are two different descriptions of an AND gate, a schematic, or netlist, description and a symbol description.

- For every simple gate you must delete the schematic description and leave the symbol description. The reason for this is that the Lattice fitter will supply its own netlist for the gates.
- There will also be a schematic and a symbol description for your main design. You need to keep the schematic and delete the symbol entry. **WARNING** As this symbol comes last in the file you must keep track of the closing brackets!!!
- As the Lattice fitter expects upper case names and EDIF is case insensitive you must add a rename clause to each simple gate as shown in bold below.

```
(cell (rename AND3 "and3") (cellType GENERIC)
(view symbol (viewType NETLIST)
(interface
(port z0 (direction OUTPUT))
(port a2 (direction INPUT))
(port a1 (direction INPUT))
(port a0 (direction INPUT))))
```

- You will also need to make sure that the simple gate descriptions come before all the others in the file. Once this is done then save the result as a .edf file (^x^w in emacs to 'save as').

It is good practice to have a look at the EDIF file and how it would be edited as this is a common requirement in industry. Fortunately for you there is a tool that will do most of what is required. Again you need to use redirection for the file input and output.

```
edi2edf <output_from_last_tool.edi >final_output_file.edf
```

Inspect this file with regard to the instructions above. The general layout should be

File header, library name etc.

```
(edif RobiN_EDIF
(edifversion 2 0 0)
(ediflevel 0)
(keywordMap (keywordLevel 0))
(status
```

```

(written
 (timeStamp 2000 03 27 10 09 59)
 (program "Driver mbk2edif")
 (author "FP & HM & OB for : ngunton")
 (dataOrigin "VLSI-CAD : Masi Lab. UPMC"))
(library alliance
 (ediflevel 0)
 (technology (numberDefinition))

```

Schematic entries for complex gates (if any were used).

```

(cell noa3_y (cellType GENERIC)
 (view schematic (viewType NETLIST)
 (interface
 (port f (direction OUTPUT))
 (port i2 (direction INPUT))
 (port i1 (direction INPUT))
 (port i0 (direction INPUT)))
 (contents
 (instance my_or1 (viewref symbol (cellref or2 (libraryref alliance))))
 (instance my_nand1 (viewref symbol (cellref nand2 (libraryref alliance))))
 (net f (joined
 (portref f )
 (portref zn0 (instanceref my_nand1 ))))
 (net i2 (joined
 (portref i2 )
 (portref a1 (instanceref my_nand1 ))))
 (net i1 (joined
 (portref i1 )
 (portref a1 (instanceref my_or1 ))))
 (net i0 (joined
 (portref i0 )
 (portref a0 (instanceref my_or1 ))))
 (net sig0 (joined
 (portref a0 (instanceref my_nand1 ))
 (portref z0 (instanceref my_or1 ))))))))

```

Symbol entries for the basic gates eg.

```

(cell( rename INV "inv") (cellType GENERIC)
 (view symbol (viewType NETLIST)
 (interface
 (port zn0 (direction OUTPUT))
 (port a0 (direction INPUT))))))

```

Schematic entry for your top level design

```

(cell microwave (cellType GENERIC)
 (view schematic (viewType NETLIST)
 (interface
 (port clk (direction INPUT))
 (port door_sw (direction INPUT))
 (port timer (direction INPUT))
 ...

```

...

symbol entry for your top level design

```
(cell microwave.lat (cellType GENERIC)
(view symbol (viewType NETLIST)
(interface
(port clk (direction INPUT))
(port door_sw (direction INPUT))
(port timer (direction INPUT))
(port t_up (direction INPUT))
(port start (direction INPUT))
(port h_pwr (direction INPUT))
(port f_pwr (direction INPUT))
(port pwr_on_rst (direction INPUT))
(port light (direction OUTPUT))
(port t_set (direction OUTPUT))
(port pwr_on (direction OUTPUT))
(port (array (rename pwr_60_1TO0_62 "pwr<1:0>") 2)(direction OUTPUT))))
)))
```

The only edits you should need to make are to move the position of the basic gates to the start of the file, after the headers and to remove the top-level symbol description of your design.

You should also look for gate descriptions that have not been modified correctly. If necessary you can hand edit your .edf file to comply with the requirements.

Phase 4:

Once you have worked through the process by hand and are comfortable with the steps required, you can use the tool `sxconv` which will automate all of the process. The `sxconv` tool require only the first input filename, your .vbe file but without the extension. eg.

```
sxconv -b myfile
```

If in doubt then read the man page for `sxconv`. This tool will convert both the behavioural subset and non-hierarchical structural subsets of VHDL. To some extent the tool relies on the correct formatting of the input file, so it may misbehave with handcrafted vbe files.

When you are satisfied that you have completed all the editing you can start on the Lattice worksheet. Remember that you will need to have saved your converted file! Your workspace/home directory is common to both the Linux system and the NT/W2K operating systems in 3P11/27/28.

Logout. If your machine has the Lattice fitter installed then reboot using CTR-ALT-DEL which will shutdown the Linux O/S correctly and restart the machine. Select NT/Windows at the boot prompt.

Now start on the Download worksheet!